Knowledge Graph Extraction from Retrieval-Augmented Generator: An Application in Aluminium Die Casting

Florian Rötzer[a], Kai Göbel[b], Maximilian Liebetreu[c] and Stephan Strommer[d]
AIT Austrian Institute of Technology GmbH, Center for Vision, Automation & Control, Giefinggasse 4, 1210 Vienna, Austria
{*florian.roetzer, kai.goebel, maximilian.liebetreu, stephan.strommer*}*@ait.ac.at*

Keywords: Knowledge Graph Extraction, Knowledge Graph Generation, Large Language Model, Retrieval-Augmented Generation, High-Pressure Die Casting.

Abstract: We present a novel, efficient, and scalable approach for generating knowledge graphs (KGs) tailored to specific competency questions, leveraging large language model (LLM)-based retrieval-augmented generation (RAG) as a source of high-quality text data. Our method utilises a predefined ontology and defines two agents: The first agent extracts entities and triplets from the text corpus maintained by the RAG, while the second agent merges similar entities based on labels and descriptions, using embedding functions and LLM reasoning. This approach does not require fine-tuning or additional AI training, and relies solely on off-the-shelf technologies. Additionally, due to the use of RAG, the method can be used with a text corpus of arbitrary size. We applied our method to the high-pressure die casting domain, focusing on defects and their causes. In the absence of annotated datasets, manual evaluation of the resulting KGs showed over 90% precision in entity extraction and around 70% precision in triplet extraction, the main source of error being the RAG itself. Our findings suggest that this method can significantly aid in the rapid generation of customised KGs for specific applications.

1 INTRODUCTION

1.1 Context and Problem

Context and Use Case. Maintaining high quality of products in high-pressure die casting (HPDC) is a challenging task that typically relies on the collaboration of experts and operators to constantly troubleshoot casting defects and ensure smooth production. Their expertise and knowledge are invaluable, and an extensive body of literature supports their efforts by detailing the correlations between defects, issues and countermeasures. However, managing the numerous factors that influence casting quality can prove difficult, especially under the time constraints of industrial production (Bonollo et al., 2015). In this context, knowledge management systems can play a pivotal role by capturing and organising expert knowledge to ensure that critical information is readily available for HPDC operations.

Assistive solutions can leverage these systems and enhance the capabilities of experts and operators. They allow them to query for specific defects and receive a list of potential causes and countermeasures, thereby streamlining the troubleshooting process. These queries for specific knowledge, as well as the expected responses, can be effectively managed using a knowledge graph (KG). KGs enable efficient querying of relationships between domain-specific concepts and entities and excel at providing answers to well-defined queries quickly and reliably. In comparison, performing the same task on a corpus of unstructured text can be challenging, confusing, and time-consuming, even for domain experts. Additionally, the symbolic and explainable nature of KGs allows for better grounding of responses compared to raw semantic text (Noy et al., 2019; Liu and Duan, 2021).

Problem: Availability of a KG. While the availability of suitable KGs varies across different domains, the extensive literature discussing the HPDC process has not yet been organised into a structured KG format, to the best of our knowledge. This circumstance also shapes the initial situation for the present work:

- A text corpus of arbitrary size is available.

[a] https://orcid.org/0000-0003-2661-5575
[b] https://orcid.org/0000-0001-5074-3652
[c] https://orcid.org/0000-0001-8374-8476
[d] https://orcid.org/0009-0009-9349-9683

Rötzer, F., Göbel, K., Liebetreu, M. and Strommer, S.
Knowledge Graph Extraction from Retrieval-Augmented Generator: An Application in Aluminium Die Casting.
DOI: 10.5220/0012951000003822

In *Proceedings of the 21st International Conference on Informatics in Control, Automation and Robotics (ICINCO 2024) - Volume 2*, pages 365-376
ISBN: 978-989-758-717-7; ISSN: 2184-2809

- A KG is needed that encompasses a comparatively small part of the corpus.
- It should decidedly not be necessary to process the corpus in its entirety.

1.2 Contribution

Contribution Goal. In this work, we aim to contribute

1. a minimal-effort approach for KG extraction from a text corpus of arbitrary size using a combination of off-the-shelf technologies, including Retrieval-Augmented Generation (RAG) and Large Language Models (LLMs),
2. applied to and evaluated for the specific application of troubleshooting in the domain of HPDC.

Challenges and Approach. We explore the use of LLM-based RAG (Gao et al., 2023) for extracting a KG tailored to troubleshooting in HPDC, given an adequately comprehensive text corpus. A significant challenge arises: the text corpus for HPDC, as well as for many other industrial domains, is too vast to be processed in its entirety by an LLM and be converted into a reasonably useful KG without any guidance. However, the portion of the text corpus relevant to the particular competency questions (CQs) that arise in HPDC troubleshooting is considerably smaller. We hypothesise that guiding the KG extraction process with CQs and a fitting ontology will yield a more complete and relevant KG, ensuring the LLM remains focused on the relevant entities and relationships and adheres to a meaningful schema.

To facilitate this, the KG extraction solution we aim for will query the Q&A interface of a RAG with prompts derived from application-specific CQs and a (rudimentary) graph ontology that provides relevant starting entities and connections to look for when parsing the RAG's responses. The technical context of this extraction task is outlined in Fig. 1. After each step of the KG extraction, we conduct a quantitative and qualitative evaluation.

Evaluation Criteria. The criteria for evaluation of generated KGs are based on confusion matrix-related scores, i.e., Precision, Recall, F1, etc. The evaluation of results can be performed

- either qualitatively, when no ground truth data is available,
- or in benchmark scenarios, that is, using annotated, complete ground truth datasets.

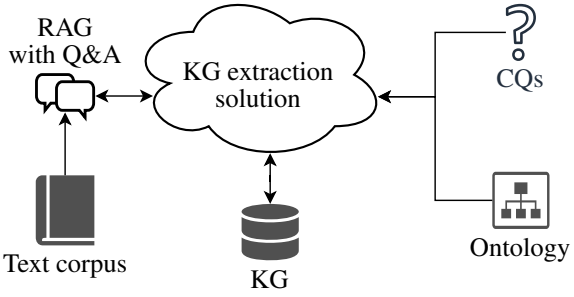

Figure 1: Context and goal: Extract an application-specific KG based on a specific ontology and CQs from a large text corpus, leveraging a RAG.

In our case, the absence of a ground truth KG allows us only to use the count of positive predictions made by the KG extraction solution. Since we aim for a comparably small and concise KG result, we will annotate the results manually with the aid of the literature and calculate the Precision score in addition to a qualitative evaluation.

1.3 Structure of this Paper

The remainder of this paper is structured as follows. Section 2 gives a short overview of the state of the art in KG extraction and other, related tasks. Section 3 contains a description of our approach and methodology, including prerequisites and assumptions about the RAG system at hand (Section 3.1), the method itself (Section 3.2), and its specific application to HPDC (Section 3.3). Section 4 provides an in-depth presentation and detailed discussion of our findings. Finally, possible implications and future avenues of research are discussed in Section 5.

2 LITERATURE REVIEW

We employ the following guidelines and structure our literature review according to Fig. 2 to apply a KG to a specific domain:

- If a suitable KG is available, utilise it.
- If only a basic KG is available, enhance or complete it.
- If no KG is available, develop one from scratch using an appropriate method.

KGs in Die Casting. For the die casting industry, a host of literature describing the process - its challenges, operation and structure - is readily available (Franke, 2019; Campbell, 2015). Pertinent problems

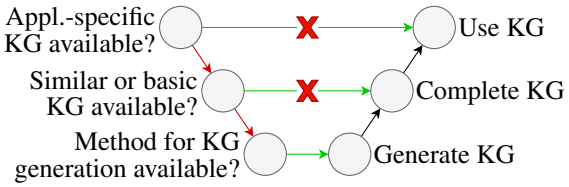

Figure 2: Guidelines for literature review.

with creating a KG from this information include (Bonollo et al., 2015)

- the variety of different sources (multiple books, reports, databases, ...),
- the language barrier (texts are written in German, English, ...),
- the ambiguous terminology (largely dependent on the authors' backgrounds and fields),
- and the lack of cross-source links (few identical entities or references across sources).

If KGs for the field of HPDC already exist, they are, to our best knowledge, not available in the public domain. Either way, such KGs may still suffer from the linguistic ambiguity problems mentioned before due to the lack of standardised terminology. For this reason, it appears instructive not to publish an individual KG but rather the method employed in its creation.

KG Fusion and Completion. One important topic in KG creation is expanding, cleaning and completing existing KGs. Active areas of research in the field include knowledge fusion (aligning entities that are sufficiently similar) (Nguyen et al., 2020; Wang et al., 2024), KG completion (predicting links and entities) (Markowitz et al., 2022; Shen et al., 2022), KG embeddings (finding embeddings to compare graphs, entities, etc.) (Biswas et al., 2023), (explainable) AI systems (recommender systems, information retrieval, question-answering systems) (Schramm et al., 2023; Purohit et al., 2020), and knowledge acquisition (extracting relations, entities and attributes) (Han et al., 2018; Al-Moslmi et al., 2020; Han and Wang, 2024).

Despite being powerful methods in their own right, both KG fusion and KG completion usually do not start triplet generation from scratch, that is, from a (near-)empty KG.

LLM for KG Completion and Reasoning. Some research has been dedicated to improving KGs using LLMs. Pan et al. (Pan et al., 2024) outline a framework for integrating LLMs with KGs, reviewing LLM-augmented KG construction techniques such as entity discovery and relation extraction. Zhang et al. (Zhang et al., 2023) investigate methods to incorporate structural information into LLMs for improved KG completion, proposing the Knowledge Prefix Adapter (KoPA) to enhance reasoning abilities. Veseli et al. (Veseli et al., 2023) evaluate the potential of language models for unsupervised knowledge base completion, offering a framework for assessing Language Models' knowledge base completion capabilities.

Further research focuses on the reasoning capabilities of LLMs on KGs. Zhang et al. (Zhang et al., 2022) discuss how to effectively fuse and reason over KG representations and language context, proposing GreaseLM, a model that integrates LLMs with graph neural networks to improve question answering by reasoning over KGs. Luo et al. (Luo et al., 2023) propose a method to enhance LLM reasoning by integrating them with KGs for more faithful and interpretable reasoning through a planning-retrieval-reasoning framework. Similarly, ReasoningLM (Jiang et al., 2023) is a model designed for subgraph reasoning over KGs in question answering tasks. For advanced reasoning, it employs a subgraph-aware self-attention mechanism and an adaptation tuning strategy.

KG Generation from Heterogeneous Sources. Parsing (semi-)structured data from the web using dedicated pages and existing links lends itself well to the creation of KGs. Similarly, parsing JSON, CSV or other structured data containers and creating KGs from them is also comparatively straightforward (Sun et al., 2024; Tamašauskaite and Groth, 2023). By contrast, retrieving entities and relations from a text corpus is a more open-ended problem, suffers from a lack of expressive metrics, and is an active area of research in natural language processing and related fields (Angelov, 2020; Lison et al., 2021; Melnyk et al., 2022; Sun et al., 2024). Additionally, the poor performance of knowledge acquisition and extraction models on PDF text has been reported, and the possible benefits of improved file parsing solutions have been highlighted (Lin, 2024).

KG Generation Involving Human Experts. A method related to the present work was developed in (Zhou et al., 2023) and applied to issues in the injection molding process. Elementary sentences in natural language were used as inputs and converted into RDF triplets by a fine-tuned BERT-based model for knowledge extraction. The extracted entities were compared to existing KG entities based on an embedding function and were merged and classified accordingly. The main drawback is that both the extraction

model and embedding function have to be fine-tuned to the technical language and dialect of the experts providing the inputs.

LMs as Knowledge Base. In recent years, LMs have gained considerable attention for their ability to access and utilise knowledge. AlKhamissi (AlKhamissi et al., 2022) discusses a direct approach to using LLMs as knowledge bases, highlighting their capabilities and limitations.

Hao et al. (Hao et al., 2023) proposed KG harvesting to extract the implicit knowledge encoded in an LM. Using BERT as the encoder, their method involves an iteration over the entire vocabulary of the LM, which, supposedly, will not scale well to LLM and RAG.

LLM for KG Extraction. Building upon the potential of LLMs as knowledge repositories, several studies have explored their application to the construction of KGs. Zhu et al. (Zhu et al., 2023) investigate the use of LLMs for KG construction and reasoning. In addition, they propose a multi-agent-based approach for constructing KGs. Carta et al. (Carta et al., 2023) present an innovative strategy for iteratively prompting LLMs to extract KG components using a zero-shot prompt strategy, thereby breaking the extraction task down into manageable steps. The method is designed to process all of the given text corpus.

Ontology Generation. Kommineni et al. (Kommineni et al., 2024) explore a semi-automatic approach that uses LLM-generated CQs along with answers from human experts to purposefully construct ontologies and KGs. Toro et al. (Toro et al., 2023) introduce the DRAGON-AI method for ontology generation, which employs LLMs and RAG to generate ontology components from existing knowledge and unstructured text. Unlike Kommineni et al., they do not provide CQs, leaving the ontology construction without a clear goal and scope.

Ideas Picked up for This Paper. In the present work, we build on the aforementioned ideas of automated KG harvesting (Hao et al., 2023) and iterative KG entity and triplet extraction (Carta et al., 2023) to achieve the envisaged contribution from Section 1.2. Section 3 covers the technical approach.

3 APPROACH

Our approach closely aligns with that of Carta et al. (Carta et al., 2023). The differences are visualised in Fig. 3. Their method processes input documents to build a KG and a hierarchical taxonomy in a bottom-up sequence. A necessary requirement is that the input documents be chosen at some point, with no way to pre-select or ignore parts of the information contained within.

Contrarily, our approach requires specific CQs to be answered and a basic ontology in which they should be answered. It therefore builds on existing domain and expert knowledge.

Our method comprises two separate routines which we call Agent 1 and Agent 2. Agent 1 directs prompts derived from our CQs at a text RAG, which serves as a library of arbitrary size. The RAG then returns short and specific text answers that are easier to process than large text documents, because they contain less unnecessary information and do not need to be split into chunks. Agent 1 then continues to extract entities and triplets for the KG. Agent 2 performs a clustering routine for KG entities, based on both embedding functions and LLM reasoning. Using separate agents with distinct responsibilities allows us to run each agent any number of times to grow and consolidate the KG.

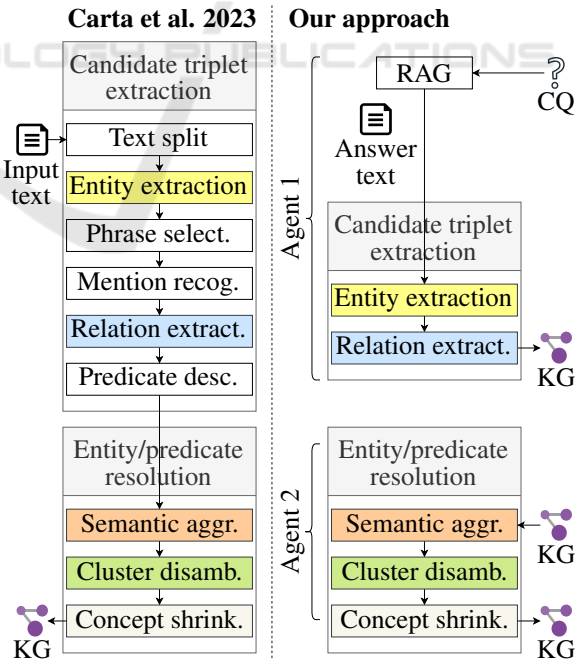

Figure 3: Comparison of our approach and the closely-related approach by Carta et al. (Carta et al., 2023)

3.1 Prerequisites

Our approach necessitates the following assumptions and prerequisites in addition to the availability of a suitable text corpus. In Section 3.3, we continue on how these prerequisites were met in our application.

RAG. Our approach relies on the availability of a RAG system with a Q&A interface for natural language. The use of a RAG allows iterative querying for small chunks of information, which drastically reduces the complexity of the KG triplet extraction task. At the same time, RAG allows us to funnel a text corpus of arbitrary size and degree of relevance into our method without compromising its performance.

CQs and Ontology. The purpose of the KG must be made explicit by stating the CQs which it should be able to answer. The way CQs are answered by the KG must be reflected by the provided ontology. From the CQs and ontology, we can then derive queries for the RAG, either automatically (Kommineni et al., 2024) or, in our case, manually.

Initial KG Entities (Optional). In most cases, the KG will not be a standalone document, but rather share well-defined interfaces with other applications, including human-machine interfaces. It is therefore reasonable to expect one or more predefined entities that might be referred to by the CQs and will serve as entry points to the KG.

3.2 Method

We split the task of KG extraction into two separate routines (cf. Fig. 3), namely Agent 1, performing candidate triplet extraction, and Agent 2, performing entity/predicate resolution.

Agent 1: Candidate Triplet Extraction from RAG. Agent 1 prompts the RAG using prompt templates derived from the provided CQs and ontology. The RAG responses are checked for minimum requirements in terms of length and relevance based on embeddings (Es et al., 2024). Sufficient answers are subsequently passed to a reduced (cf. Fig. 3) triplet extraction pipeline leveraging an LLM prompt for KG triplets, derived from the CQ to be answered. The extracted triplets are then written back to the KG. Figure 4 shows a flowchart of this procedure.

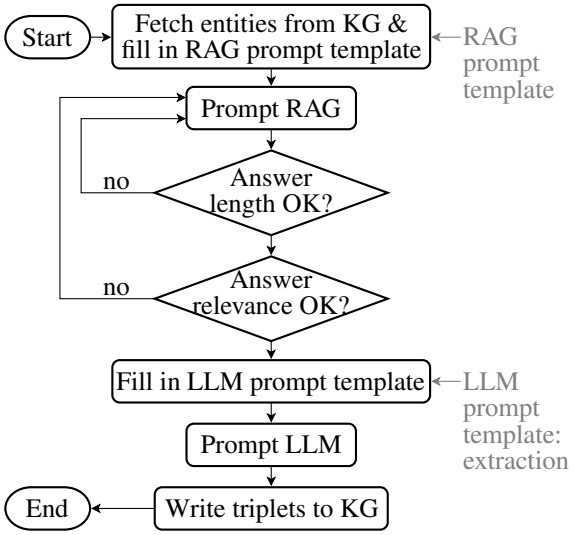

Figure 4: Flow chart of Agent 1; From RAG to KG triplets.

Agent 2: Entity Resolution. Agent 2 takes samples of entities from the KG and merges sufficiently similar ones. To this end, each entity is required to have a `label` and a `description` property. The procedure itself is twofold (excluding the operations on the KG) and is outlined in Fig. 5.

First (semantic aggregation), the `label` and `description` of each entity are mapped to an embedding space by means of the Universal Sentence Encoder (Cer et al., 2018). The embedding utilises the cosine distance for similarity of sentences, which is not a metric in the mathematical sense. We therefore only rely on the closeness of two entities at a time, thereby finding single-linkage hierarchical clustering to be a reasonable choice. This step yields clusters that form chains in the embedding space from one concept to another.

In the second step (cluster disambiguation + concept shrinkage), we present an LLM with the embedding-based clusters and query it to further partition them if necessary. The LLM also assigns a unifying `label` and `description` to each cluster with two or more elements.

3.3 Application in HPDC

For our application in HPDC, we generate a KG that links casting defects with the issues that cause them. The purpose of such a graph is to point out the correlations and dependencies between different defect patterns, which are important to understand when taking countermeasures against occurring defects.

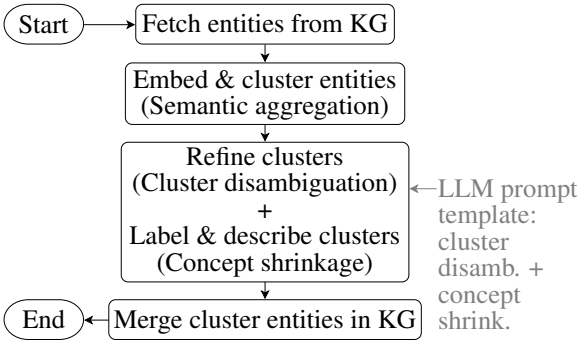

Figure 5: Flow chart of Agent 2: From KG entities to merged similar KG entities.

3.3.1 Prerequisites

We first describe how we achieve the prerequisites described in Section 3.1 in the context of our HPDC application.

RAG. We refrain from developing our own RAG system and instead rely on Perplexity. Our expectation here is that it represents the state of the art in RAG technology, both in terms of text corpus size and well-tuned parameterisation.

CQ and Ontology. We utilise but a single CQ to make the results easily tractable: "What are possible issues that cause the defect XY?" Likewise, the ontology we use can be reduced to `(Issue)-[causes]->(Defect)` in Cypher notation. From the CQ and ontology, we derive a prompt template to query the RAG for `Issue` entities that cause a specific `Defect`.

Initial KG Entities. We provide `Defect` entities based on the norm CEN/TR 16749 (European Committee for Standardization, Technical Committee 132, 2014) for aluminium casting defects.[1] We use the defect names, definitions and morphologies provided in the norm document as `label` and `description` properties, respectively.

3.3.2 Implementation

We implemented the agents described in Section 3.2 in Python. The utilised technologies and their parameterisation are listed in the following.

[1]The norm itself is not freely available. However, the casting defect definitions therein are derived from the reports of the project The MUSIC guide to key-parameters in High Pressure Die Casting

Implementation of Agent 1. A comprehensive list of the used technologies and parameters is given in Tables 1 and 2.

Table 1: Technologies used for Agent 1.

KG	Neo4j
RAG	Perplexity
Embedding model	text-embedding-ada-002
LLM	gpt-3.5-turbo

Table 2: Parameterisation used for Agent 1.

RAG temperature	Perplexity default
RAG ans. min. length	100 Tokens
RAG ans. min. relevance	0.9
LLM temperature	0

Listing 1: RAG prompt template.

```
Read the following description of
    ↪ {defect[label]} in the context
    ↪ of aluminium high-pressure die
    ↪ casting defects. Find possible
    ↪ issues that cause this defect.
    ↪ Give a detailed answer.

Description:
{defect[description]}
```

Listing 2: LLM prompt template for triplet extraction.

```
Read and analyse the context given
    ↪ below. What are possible issues
    ↪  that cause {defect[label]}?
Rewrite the description of each issue
    ↪  to focus on the underlying
    ↪ phenomenon. Return all causes
    ↪ with their descriptions in a
    ↪ JSON structure of the following
    ↪  form:
[{{"name":"...","description
    ↪ ":"..."}},...]
If no causing issues can be inferred
    ↪ from the context, return [].

Context:
{answer}

Answer:
```

Implementation of Agent 2. A comprehensive list of the used technologies and parameters is given in Tables 3 and 4. Note that the cluster distance threshold for hierarchical clustering should be chosen depending on the linkage dendrogram of the entity set that is being clustered (see Section 4). We choose this

value such that the largest cluster will not exceed the context length of the LLM.

Table 3: Technologies used for Agent 2.

KG	Neo4j
Embedding model	Universal Sentence Encoder
LLM	gpt-4o

Table 4: Parameterisation used for Agent 2.

Embedding distance	Cosine
Cluster linkage	Single
Cluster distance threshold	0...2
LLM temperature	0.2
LLM top_p	0.1

Listing 3: LLM prompt template for cluster disambiguation.

```
You are an intelligent clustering
    ↪ algorithm. You will be provided
    ↪  with a JSON-like string
    ↪ representing a cluster with its
    ↪  members. Analyse the data and
    ↪ identify different categories
    ↪ or clusters within the provided
    ↪  data. Split the data into
    ↪ appropriate clusters if needed
    ↪ and provide the output as a
    ↪ JSON string with the new
    ↪ clusters.

The output will follow the pattern:
[
{
    "label": "Cluster Name",
    "description": "Cluster
        ↪ Description.",
    "members": [
        {
            "id": "unique_id",
            "label": "Label Name",
            "description": "
                ↪ Description of the
                ↪ label."
        },
        ...
    ]
},
...
]

Input:
{cluster}

Output:
```

4 RESULTS AND DISCUSSION

The agents from Section 3 can be run in multiple iterations to continuously grow and extend the KG until it converges in size. However, in order to keep the results of this paper reproducible and comprehensible, we show and discuss the KG we acquired after running both Agent 1 and Agent 2 once. We annotate the results manually, using the norm definitions of each casting defect (European Committee for Standardization, Technical Committee 132, 2014).

An example result with annotation is shown in Table 5. The table shows the labels of some `Issue` entites extracted by Agent 1 for different `Defects` - in this case "Interdendritic shrinkage" (A1.2) and "Air entrapment porosity" (A2.1). We differentiate between the validity of extracted entities in the sense of the ontology and application, and the validity of triplets, i.e., relationships of the form `(Issue)-[causes]->(Defect)`. The example shows four extracted entities in the correct triplet and one correct entity in the wrong triplet. Furthermore, the `Issue` entities "Alloy composition" and "Poor metal quality" may be semantically similar enough to later be clustered by Agent 2.

Table 5: Example result from Agent 1 for the `Defects` "Interdendritic shrinkage" (A1.2) and "Air entrapment porosity" (A2.1); Each row can be true (T) or false (F) either as an entity (Ent.) or as a triplet (Tri.).

Defect ID	Issue	Ent.	Tri.
...	...	...	...
A1.2	Inadequ. liquid flow	T	T
A1.2	High solidific. rate	T	T
A1.2	Alloy composition	T	T
A1.2	Cooling rate	T	T
...	...	...	...
A2.1	Poor metal quality	T	F
...	...	...	...

There is no ground truth available by which the results can be evaluated. Therefore, we cannot compute a complete confusion matrix from the results. However, we can evaluate the Precision metric P as

$$P = \frac{\mathrm{TP}}{\mathrm{PP}} \tag{1}$$

from the number of true-positives TP and total predicted positives PP. This score indicates what portion of the extracted and clustered entities and triplets is correct.

4.1 Numerical Results

The numerical results will be displayed in the form shown in Fig. 6. The incidence matrix in the middle visualises the clustering result. The x-ticks represent the extracted `Issue` entities and the y-ticks represent the clusters defined by Agent 2. If only Agent 1 was used, the clustering result is simply the identity matrix.

The vector on the left visualises the Precision score P for the extracted entities or clusters, respectively. The green and red tiles thereby represent true and false positive predictions of `Issue` entities or clusters, i.e., whether an extracted entity or cluster is actually an `Issue` in the sense of the ontology. In this way, the entity extraction Precision is made visible.

The incidence matrix on the right side of Fig. 6 visualises the extracted triplets of the form `(Issue)-[causes]->(Defect)`. The green and red tiles in the right matrix represent true and false positive predictions for triplets, respectively. The right matrix thereby visualises the triplet extraction Precision.

4.2 Agent 1 Results

Experiment 1. We start by testing Agent 1 alone. The outcome is shown in Figure 6. Since at this point no clustering was performed, all found `Issue` entities (and corresponding triplets) are separated and each one is a cluster of its own. The result in Fig. 6 separately visualises the Precision of Agent 1 in extracting entities that fit the ontology definition and triplets that are actually true.

The shown results correspond to a Precision of $P = 90.85\%$ in the extraction of singular `Issues`, but only $P = 68.31\%$ in triplet extraction. This means that Agent 1, together with Perplexity, was able to find and correctly describe issues that generally appear in HPDC with $P > 90\%$. However, the Precision of triplet extraction at only $P = 68.31\%$ hints that the `Issues` were not always connected to the right `Defects` by the RAG.

The reason for this drop in Precision can be located either in the way the RAG prompt was formulated, or in the used RAG technology itself. Possible explanations include:

- Poor prompting of the RAG, or an abundance of information in the prompt, both misleading the RAG on what to retrieve.
- The RAG not correctly processing the provided descriptions.
- No norm-conforming technical language in the source documents.

The score P for triplet extraction can essentially be improved from two sides.

1. Either the RAG interaction step is improved, which involves prompt engineering and RAG development, or
2. a way to block out wrong information after triplet extraction is implemented, which would again involve human expert interaction at some point.

In the scope of this work, we refrain from further engineering on Agent 1 and proceed to Agent 2 with the acquired results.

4.3 Agent 2 Results

Unlike Agent 1, according to Table 4, Agent 2 has an obvious degree of freedom in its parameterisation: the cluster distance threshold. For this reason, we perform two experiments with different threshold settings for Agent 2.

Figure 7 shows the single-linkage clustering dendrogram of the results from Experiment 1, based on which we will choose different values for the cluster distance threshold.

Experiment 2A: Cluster Threshold $= 1.133$. In accordance with Section 3.2, we chose the cluster threshold such that the largest cluster has 16 elements, in order to not overstrain the attention span of the LLM. The results of Agent 2 in succession of Agent 1 are shown in Fig. 8. The entity precision score of $P = 90.59\%$ refers to `Issue` clusters conforming to the ontology.

Since this is only a partitioning step, the entity precision is essentially the same as in Experiment 1. However, we did notice a significant reduction in specificity in the entity descriptions. As an example, Table 6 shows an `Issue` cluster "Temperature issues", found by Agent 2. Along with this decreased specificity of language, it is not surprising that the Precision score of the triplet extraction slightly increased to $P = 70.07\%$.

Table 6: Example result from Agent 1+2: `Issue` entities clustered together by Agent 2 as "Temperature issues".

`Defect` ID	Original `Issue`	Cluster
A2.4	Die temperature	Temp. iss.
A5.1	Temperature factors	Temp. iss.
B2.1	High die temperature	Temp. iss.
B5.1	Temperature factors	Temp. iss.

Diluting the descriptions of clusters of `Issues` seems beneficial from the perspective of the LLM. However, it also requires more knowledge on the side

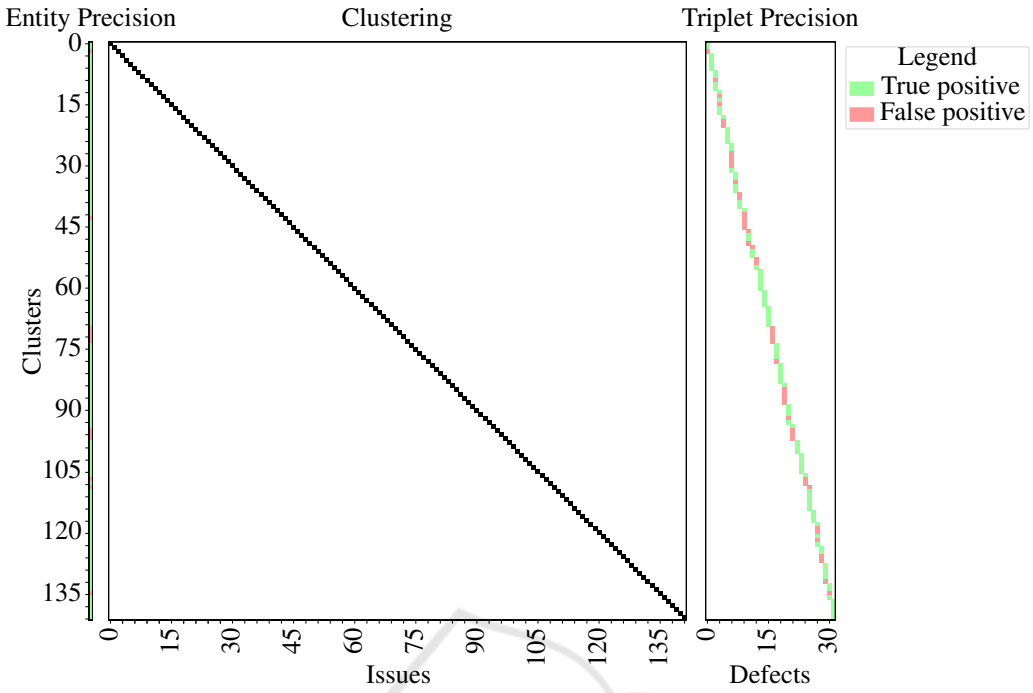

Figure 6: Experiment 1 result plot; Vector on the left: Precision of correct `Issue` entity extraction (90.85%); Left matrix: Cluster incidence matrix (Identity, because Agent 2 was not used at this point); Right matrix: Triplet incidence matrix and Precision of correct triplet extraction (68.31%).

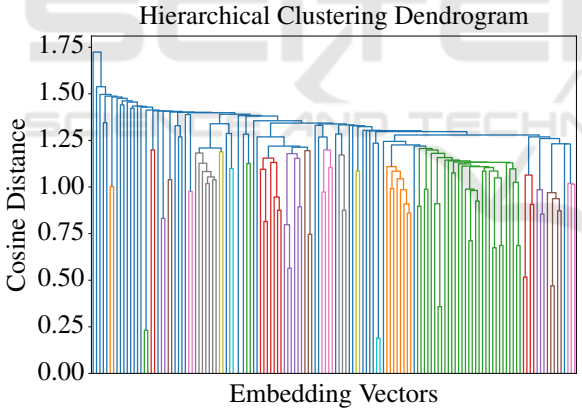

Figure 7: Single-linkage clustering dendrogram of the results from Experiment 1.

of the user to understand what exactly the KG entities are referring to. If this dilusion of language becomes problematic, one might have to rethink the clustering process or enforce the use of a fixed vocabulary set. At any rate, as this task would involve user feedback, it remains out of scope for the present paper.

Experiment 2B: Cluster Threshold $= 1.21$**.** We now choose the cluster threshold such that the largset cluster passed to the LLM has 32 elements. The result of Agents 1+2 is shown in Fig. 9. We notice that after the LLM step of Agent 2, the number of clusters is only slightly different from Experiment 2A. Also, in terms of Precision, the extraction of `Issue` clusters with $P = 87.5\%$ and triplets of the form with $P = 72.93\%$ both stayed roughly the same. The slight drop in entity extraction precision is owed to an overly strong dilusion in technical language. In fact, one cluster was only labelled ”Casting Defects”, which we had to annotate as wrong because it was stripped of all useful information.

A visual comparison of the clustering matrices from experiments A and B suggests that the clustering process is relatively stable, also with the LLM involved. However, the linguistic dilusion caused by a strong LLM involvement needs to be addressed in future works.

4.4 Discussion

We performed three experiments (Experiments 1, 2A and 2B) with the implemented Agents 1 and 2. The outcomes show and confirm, in accordance with other results from the literature, that the bottleneck of this approach as well as other natural language processing applications is the variety and liberal use of technical vocabulary in HPDC, just like in other fields.

As for the extraction task covered by Agent 1, our approach relies heavily on the interaction with RAG

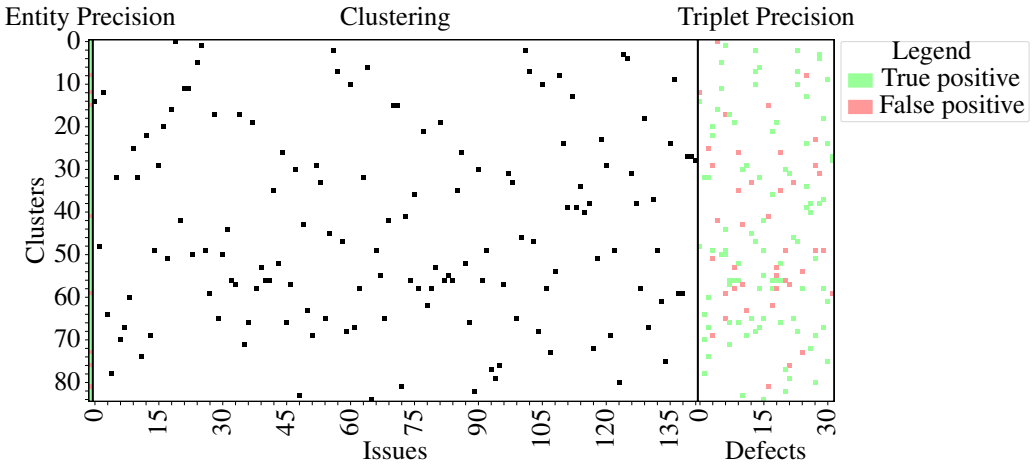

Figure 8: Experiment 2A result plot; Vector: Precision of correct `Issue` cluster extraction (90.59%); Left matrix: Clustering; Right matrix: Triplet extraction and Precision (70.07%).

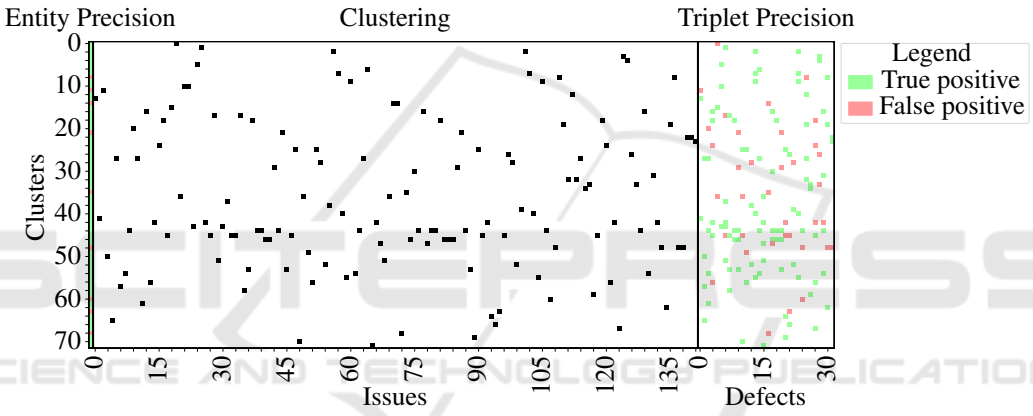

Figure 9: Experiment 2B result plot; Vector: Precision of correct `Issue` cluster extraction (87.50%); Left matrix: Clustering; Right matrix: Triplet extraction and Precision (72.93%).

or, more generally, a Q&A interface. Inconsistencies in this interaction regarding the handling of technical vocabulary have near-indelible consequences on the Precision and, in consequence, the reliability of the results. Future works building on our results should, to some extent, take the technical vocabulary (taxonomy) of the respective field of application into account, which is also a fairly common step when constructing KGs and ontologies.

The clustering task covered by Agent 2 has proven to be relatively robust. However, the description of clustered entities becomes strongly diluted, especially when the clustering is not really appropriate. Further efforts in terms of prompt engineering may help keep up the linguistic precision, but without a fixed, compulsory vocabulary to adhere to, the clustering LLM will likely dilute the entity or cluster contents over time.

5 CONCLUSIONS

Contribution Review. In this paper, we introduced a novel, simplified, and scalable approach for KG extraction from a text corpus equipped with RAG. The method is supplied with specific CQs and a predefined ontology that dictates the answer format to those CQs. It leverages a state-of-the-art RAG (Perplexity) and comprises two agents:

- Agent 1 extracts entities and triplets from the answers provided by the RAG.
- Agent 2 merges sufficiently similar entities by comparison of label and description properties.

Our method features the following advantages over other approaches:

- Due to the use of RAG, the underlying text corpus may be of arbitrary size.

- The method refrains from fine-tuning or other AI training and uses only off-the-shelf technologies.

We tested our approach in an application for troubleshooting in HPDC. A list of possible defects was given as a starting point. Due to the absence of annotated datasets in this domain, we evaluated our approach by manually annotating the resulting KG using a norm for casting defect classification. We then assessed Precision scores for both entities and triplets. Our method achieved over 90% Precision in entity extraction, indicating exceptional performance in this area. The few extraction errors were mainly attributed to inaccuracies in the description texts provided by the RAG. Triplet extraction Precision was around 70%, suggesting that some entities were incorrectly placed in the RAG answers.

Industrial Impact. The results of this work are only one example of how automated KG extraction could be applied to various industrial domains. The ability to generate tailored KGs allows for the customisation of knowledge bases to address specific quality concerns in HPDC. This may enable more targeted and effective troubleshooting, thereby reducing downtime and improving overall manufacturing quality. The method's scalability also means it can be adapted to various CQs, providing a flexible and promising tool for continuous quality improvement in HPDC and potentially other manufacturing domains.

Future Work. Several aspects of our solution have not yet been covered and will be addressed in future works. Assuming sufficient domain knowledge to cover an appropriate CQ and create a fully annotated dataset as groud truth, the evaluation of the KG result could be extended to a full confusion matrix. Additionally, multiple iterations of KG extraction can cover for the influence of randomized algorithms in the RAG and LLM systems. We expect that the KG result will converge in size and quality - however, it is difficult to predict what such a steady state may look like. Lastly, the generation of RAG and LLM queries from the given CQs and ontology should be automated to complete the KG extraction pipeline.

ACKNOWLEDGEMENTS

This work was conducted within the Austrian research project DG Assist (FFG project number: FO999899053). This project is funded by the Federal Ministry for Climate Protection, Environment, Energy, Mobility, Innovation and Technology, BMK, and is carried out as part of the Production of the Future programme.

OpenAI's GPT-4o model (OpenAI, 2024) was used to aid in the formulation of the Abstract. DeepL Write (DeepL, 2024) was used to convert the text from American to British English.

REFERENCES

Al-Moslmi, T., Ocaña, M. G., Opdahl, A. L., and Veres, C. (2020). Named entity extraction for knowledge graphs: A literature overview. *IEEE Access*, 8:32862–32881.

AlKhamissi, B., Li, M., Celikyilmaz, A., Diab, M., and Ghazvininejad, M. (2022). A review on language models as knowledge bases. *arXiv preprint arXiv:2204.06031*.

Angelov, D. (2020). Top2vec: Distributed representations of topics. *arXiv preprint arXiv:2008.09470*.

Biswas, R., Kaffee, L.-A., Cochez, M., Dumbrava, S., Jendal, T. E., Lissandrini, M., Lopez, V., Mencía, E. L., Paulheim, H., Sack, H., Vakaj, E. K., and de Melo, G. (2023). Knowledge Graph Embeddings: Open Challenges and Opportunities. *Transactions on Graph Data and Knowledge*, 1(1):4:1–4:32.

Bonollo, F., Gramegna, N., and Timelli, G. (2015). High-pressure die-casting: Contradictions and challenges. *JOM*, 67(5):901–908.

Campbell, J. (2015). *Complete Casting Handbook*. Elsevier Science & Technology, Oxford, 2nd ed. edition.

Carta, S., Giuliani, A., Piano, L., Podda, A. S., Pompianu, L., and Tiddia, S. G. (2023). Iterative zero-shot llm prompting for knowledge graph construction. *arXiv preprint arXiv:2307.01128*.

Cer, D., Yang, Y., Kong, S., Hua, N., Limtiaco, N., John, R. S., Constant, N., Guajardo-Cespedes, M., Yuan, S., Tar, C., Sung, Y., Strope, B., and Kurzweil, R. (2018). Universal sentence encoder. *CoRR*, abs/1803.11175.

DeepL (2024). Deepl write. Online: https://www.deepl.com/en/write (accessed 2024-06-24).

Es, S., James, J., Anke, L. E., and Schockaert, S. (2024). Ragas: Automated evaluation of retrieval augmented generation. In Aletras, N. and Clercq, O. D., editors, *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2024 - System Demonstrations, St. Julians, Malta, March 17-22, 2024*, pages 150–158. Association for Computational Linguistics.

European Committee for Standardization, Technical Committee 132 (2014). CEN/TR 16749.

Franke, S. (2019). *Giesserei-Lexikon*. Schiele & Schön, 20th edition.

Gao, Y., Xiong, Y., Gao, X., Jia, K., Pan, J., Bi, Y., Dai, Y., Sun, J., Guo, Q., Wang, M., and Wang, H. (2023). Retrieval-augmented generation for large language models: A survey. *CoRR*, abs/2312.10997.

Han, X., Liu, Z., and Sun, M. (2018). Neural knowledge acquisition via mutual attention between knowledge

graph and text. *Proceedings of the AAAI Conference on Artificial Intelligence*, 32(1).

Han, Z. and Wang, J. (2024). Knowledge enhanced graph inference network based entity-relation extraction and knowledge graph construction for industrial domain. *Frontiers of Engineering Management*, 11(1):143–158.

Hao, S., Tan, B., Tang, K., Ni, B., Shao, X., Zhang, H., Xing, E. P., and Hu, Z. (2023). Bertnet: Harvesting knowledge graphs with arbitrary relations from pretrained language models. In Rogers, A., Boyd-Graber, J. L., and Okazaki, N., editors, *Findings of the Association for Computational Linguistics: ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 5000–5015. Association for Computational Linguistics.

Jiang, J., Zhou, K., Zhao, W. X., Li, Y., and Wen, J.-R. (2023). Reasoninglm: Enabling structural subgraph reasoning in pre-trained language models for question answering over knowledge graph. *arXiv preprint arXiv:2401.00158*.

Kommineni, V. K., König-Ries, B., and Samuel, S. (2024). From human experts to machines: An llm supported approach to ontology and knowledge graph construction. *arXiv preprint arXiv:2403.08345*.

Lin, D. (2024). Revolutionizing retrieval-augmented generation with enhanced pdf structure recognition. *arXiv preprint arXiv:2401.12599*.

Lison, P., Barnes, J., and Hubin, A. (2021). skweak: Weak supervision made easy for NLP. In Ji, H., Park, J. C., and Xia, R., editors, *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing: System Demonstrations*, pages 337–346, Online. Association for Computational Linguistics.

Liu, J. and Duan, L. (2021). A survey on knowledge graph-based recommender systems. In *2021 IEEE 5th Advanced Information Technology, Electronic and Automation Control Conference (IAEAC)*, pages 2450–2453.

Luo, L., Li, Y.-F., Haffari, G., and Pan, S. (2023). Reasoning on graphs: Faithful and interpretable large language model reasoning. *arXiv preprint arXiv:2310.01061*.

Markowitz, E., Balasubramanian, K., Mirtaheri, M., Annavaram, M., Galstyan, A., and Ver Steeg, G. (2022). StATIK: Structure and text for inductive knowledge graph completion. In Carpuat, M., de Marneffe, M.-C., and Meza Ruiz, I. V., editors, *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 604–615, Seattle, United States. Association for Computational Linguistics.

Melnyk, I., Dognin, P., and Das, P. (2022). Knowledge graph generation from text. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 1610–1622, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Nguyen, H. L., Vu, D. T., and Jung, J. J. (2020). Knowledge graph fusion for smart systems: A survey. *Information Fusion*, 61:56–70.

Noy, N., Gao, Y., Jain, A., Narayanan, A., Patterson, A., and Taylor, J. (2019). Industry-scale knowledge graphs: lessons and challenges. *Commun. ACM*, 62(8):36–43.

OpenAI (2024). Hello gpt-4o. Online: https://openai.com/index/hello-gpt-4o (accessed 2024-06-24).

Pan, S., Luo, L., Wang, Y., Chen, C., Wang, J., and Wu, X. (2024). Unifying large language models and knowledge graphs: A roadmap. *IEEE Transactions on Knowledge and Data Engineering*.

Purohit, H., Shalin, V. L., and Sheth, A. P. (2020). Knowledge graphs to empower humanity-inspired ai systems. *IEEE Internet Computing*, 24(4):48–54.

Schramm, S., Wehner, C., and Schmid, U. (2023). Comprehensible artificial intelligence on knowledge graphs: A survey. *Journal of Web Semantics*, 79:100806.

Shen, T., Zhang, F., and Cheng, J. (2022). A comprehensive overview of knowledge graph completion. *Knowledge-Based Systems*, 255:109597.

Sun, Q., Luo, Y., Zhang, W., Li, S., Li, J., Niu, K., Kong, X., and Liu, W. (2024). Docs2kg: Unified knowledge graph construction from heterogeneous documents assisted by large language models. *arXiv preprint arXiv:2406.02962*.

Tamašauskaite, G. and Groth, P. (2023). Defining a knowledge graph development process through a systematic review. *ACM Trans. Softw. Eng. Methodol.*, 32(1).

Toro, S., Anagnostopoulos, A. V., Bello, S., Blumberg, K., Cameron, R., Carmody, L., Diehl, A. D., Dooley, D., Duncan, W., Fey, P., et al. (2023). Dynamic retrieval augmented generation of ontologies using artificial intelligence (dragon-ai). *arXiv preprint arXiv:2312.10904*.

Veseli, B., Singhania, S., Razniewski, S., and Weikum, G. (2023). Evaluating language models for knowledge base completion. In *European Semantic Web Conference*, pages 227–243. Springer.

Wang, Y., Wan, Y., Bai, L., Cui, L., Xu, Z., Li, M., Yu, P. S., and Hancock, E. R. (2024). Collaborative knowledge graph fusion by exploiting the open corpus. *IEEE Transactions on Knowledge and Data Engineering*, 36(2):475–489.

Zhang, X., Bosselut, A., Yasunaga, M., Ren, H., Liang, P., Manning, C. D., and Leskovec, J. (2022). Greaselm: Graph reasoning enhanced language models for question answering. *arXiv preprint arXiv:2201.08860*.

Zhang, Y., Chen, Z., Zhang, W., and Chen, H. (2023). Making large language models perform better in knowledge graph completion. *arXiv preprint arXiv:2310.06671*.

Zhou, Z.-W., Ting, Y.-H., Jong, W.-R., Chen, S.-C., and Chiu, M.-C. (2023). Development and application of knowledge graphs for the injection molding process. *Machines*, 11(2):271.

Zhu, Y., Wang, X., Chen, J., Qiao, S., Ou, Y., Yao, Y., Deng, S., Chen, H., and Zhang, N. (2023). Llms for knowledge graph construction and reasoning: Recent capabilities and future opportunities. *arXiv preprint arXiv:2305.13168*.